ProSeminar „Mobilfunksysteme“

Vortrag von Stefan Büttcher

Background und Philosophie

Background und Philosophie

- initiiert 1997 von Motorola, Nokia, Ericsson und Phone.com;

Background und Philosophie

- initiiert 1997 von Motorola, Nokia, Ericsson und Phone.com;
- erster Versuch, Internet-Zugriff für mobile Kommunikation zu standardisieren;

Background und Philosophie

- initiiert 1997 von Motorola, Nokia, Ericsson und Phone.com;
- erster Versuch, Internet-Zugriff für mobile Kommunikation zu standardisieren;
- Client/Server-basiertes System;

Background und Philosophie

- initiiert 1997 von Motorola, Nokia, Ericsson und Phone.com;
- erster Versuch, Internet-Zugriff für mobile Kommunikation zu standardisieren;
- Client/Server-basiertes System;
- konzipiert für den Einsatz

Background und Philosophie

- initiiert 1997 von Motorola, Nokia, Ericsson und Phone.com;
- erster Versuch, Internet-Zugriff für mobile Kommunikation zu standardisieren;
- Client/Server-basiertes System;
- konzipiert für den Einsatz
    - in jedem beliebigen Mobiltelefon,

Background und Philosophie

- initiiert 1997 von Motorola, Nokia, Ericsson und Phone.com;
- erster Versuch, Internet-Zugriff für mobile Kommunikation zu standardisieren;
- Client/Server-basiertes System;
- konzipiert für den Einsatz
  - in jedem beliebigen Mobiltelefon,
  - auf jedem beliebigen Wireless Service (z.B. SMS, USSD, GPRS),

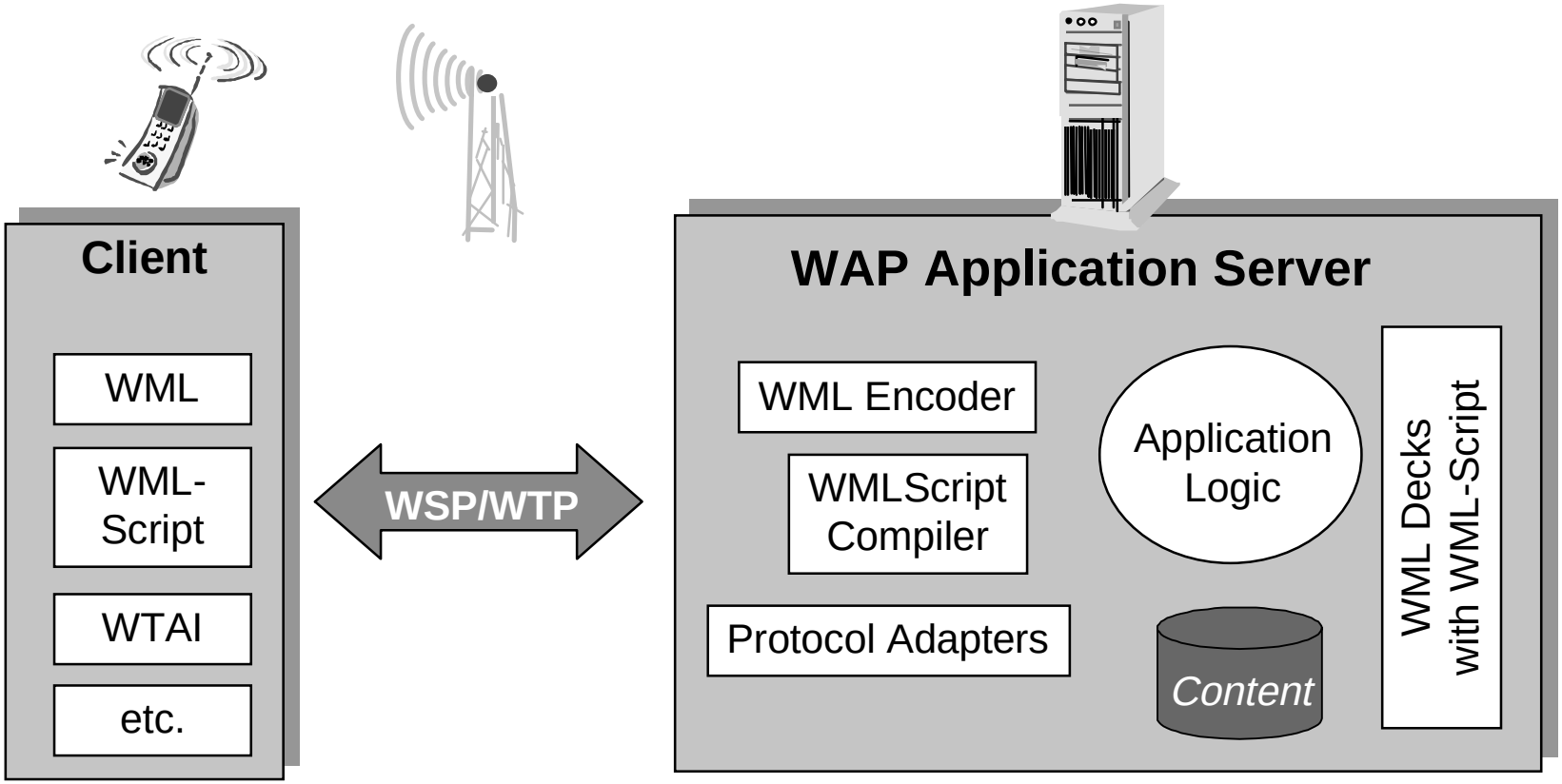
Background und Philosophie

- initiiert 1997 von Motorola, Nokia, Ericsson und Phone.com;
- erster Versuch, Internet-Zugriff für mobile Kommunikation zu standardisieren;
- Client/Server-basiertes System;
- konzipiert für den Einsatz
  - in jedem beliebigen Mobiltelefon,
  - auf jedem beliebigen Wireless Service (z.B. SMS, USSD, GPRS),
  - mit jedem beliebigen Mobilfunk-Standard (z.B. GSM, UMTS),

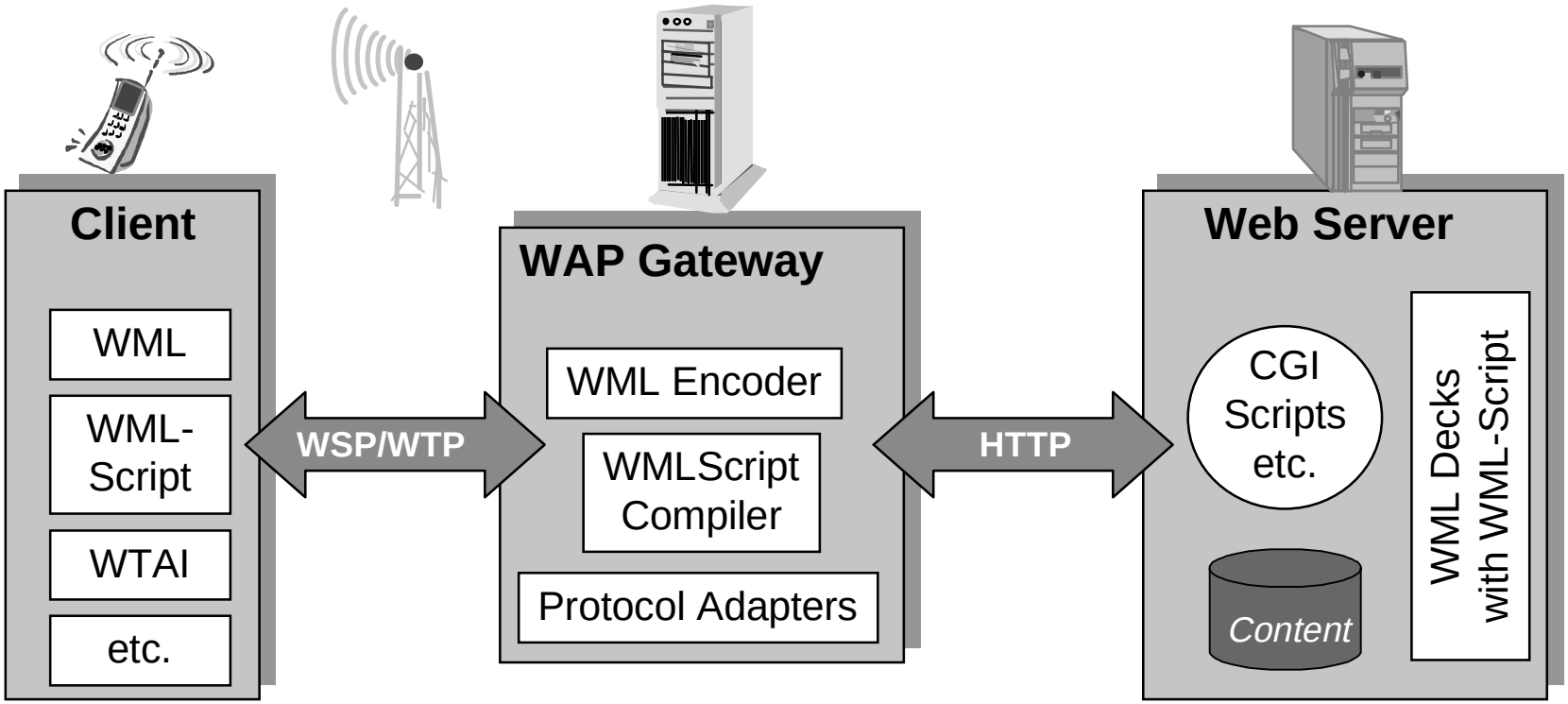
Background und Philosophie

- initiiert 1997 von Motorola, Nokia, Ericsson und Phone.com;
- erster Versuch, Internet-Zugriff für mobile Kommunikation zu standardisieren;
- Client/Server-basiertes System;
- konzipiert für den Einsatz
  - in jedem beliebigen Mobiltelefon,
  - auf jedem beliebigen Wireless Service (z.B. SMS, USSD, GPRS),
  - mit jedem beliebigen Mobilfunk-Standard (z.B. GSM, UMTS),
  - mit jedem beliebigen Eingabemedium (z.B. Keyboard, Touch-Screen).

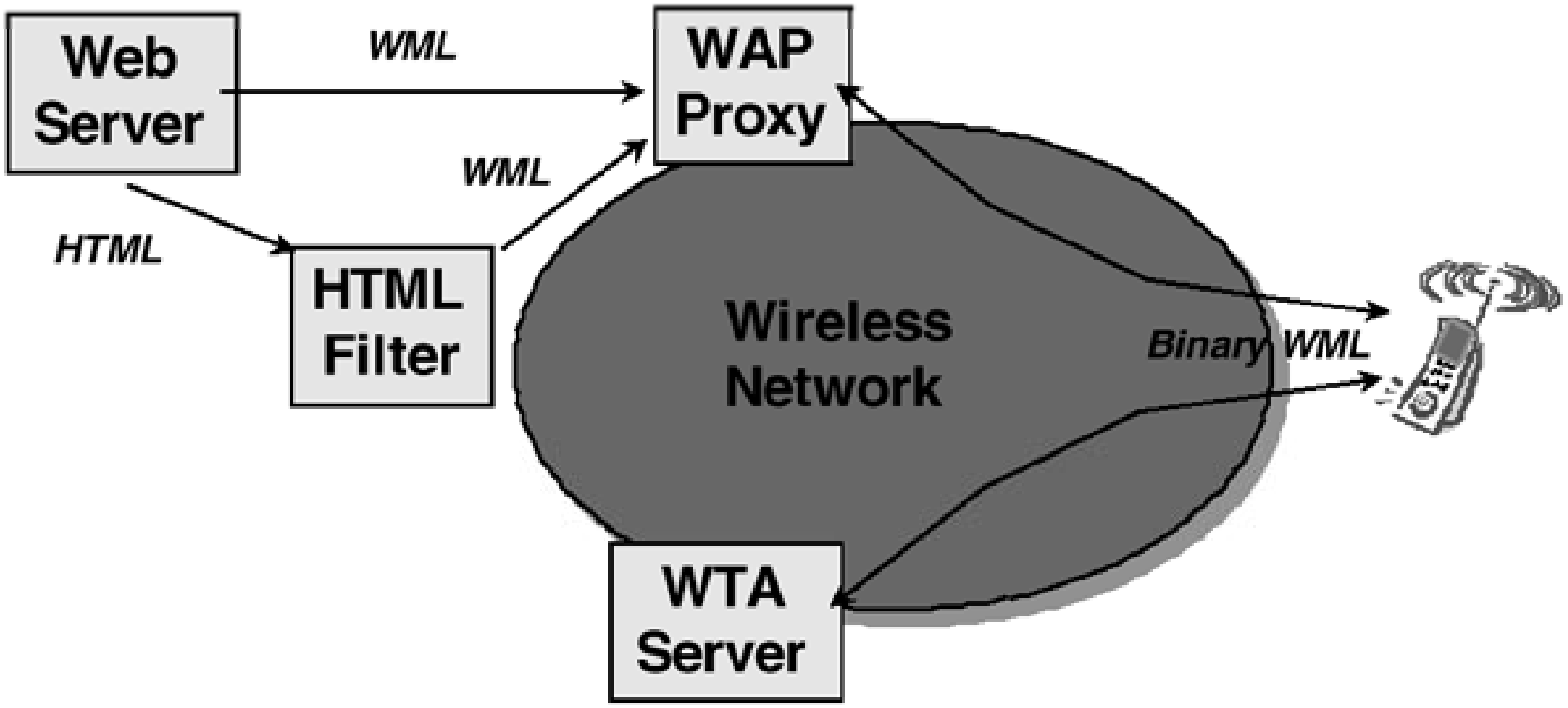
erlick Prchiekur



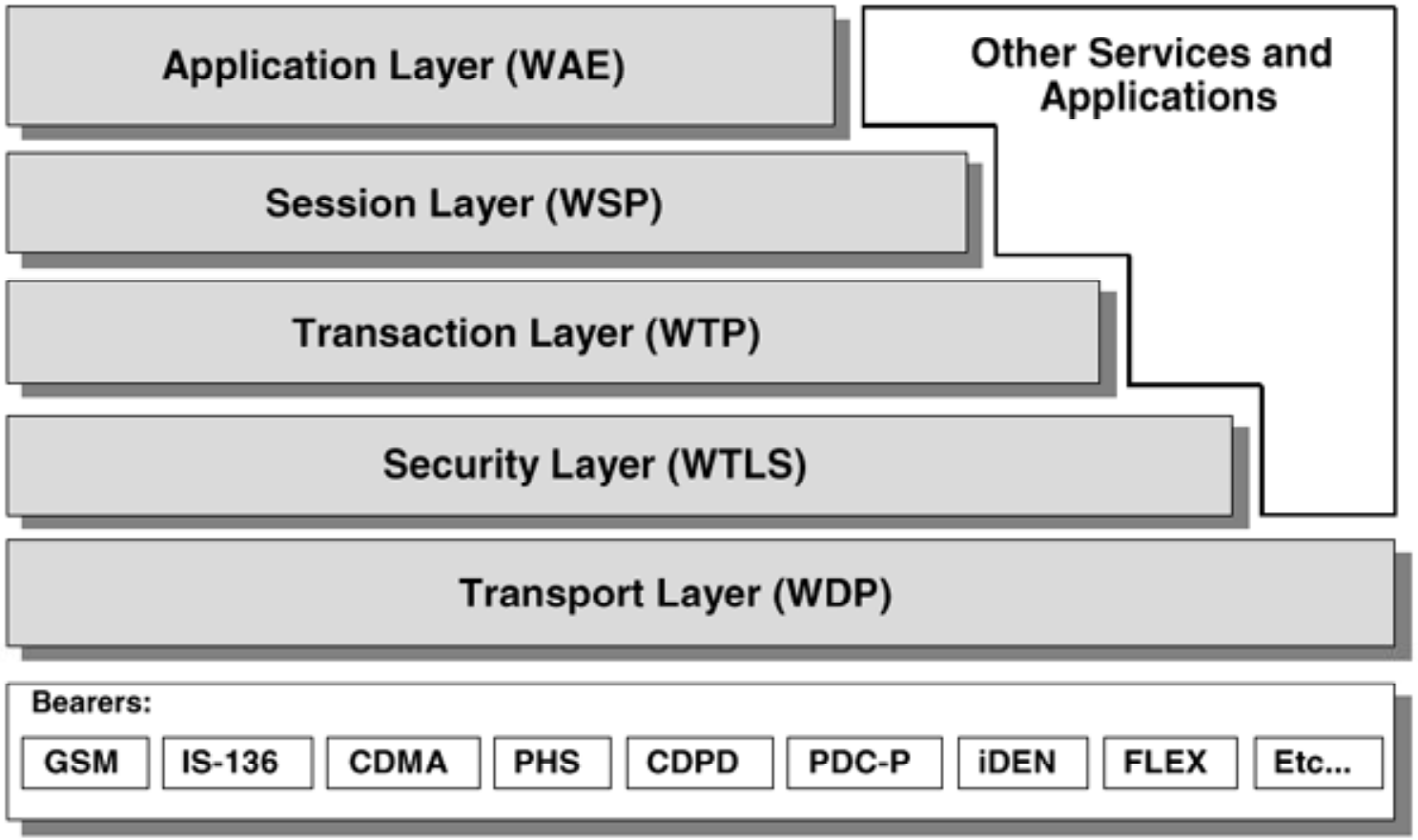
erlick Prchiekur



erlick Prchiekur



P Proocol ack

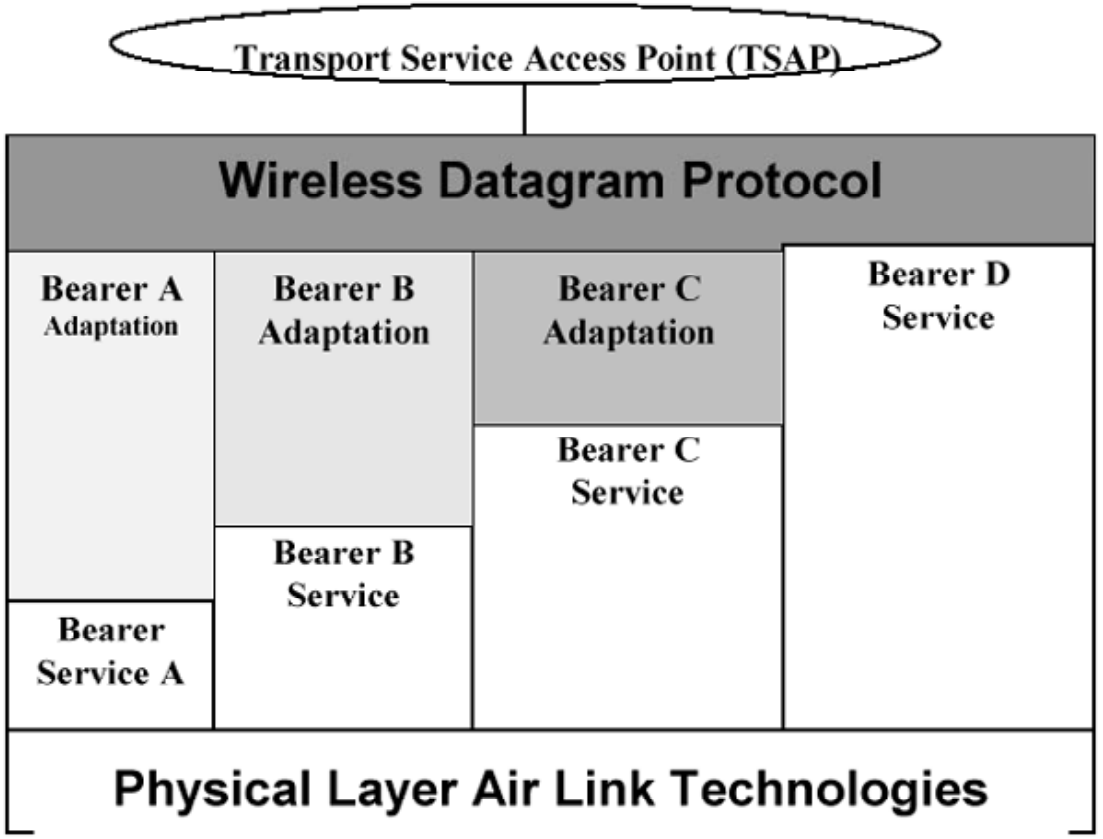


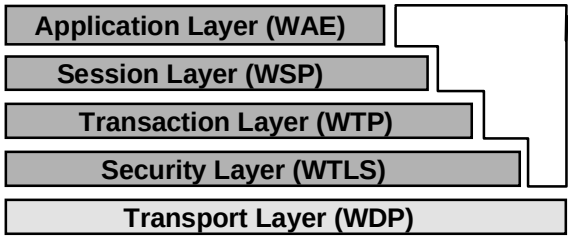
ireless aagra Proocol

Mögliche Trägerdienste sind z.B. SMS, USSD, CSD und GPRS.

Das jeweilige Gegenüber wird dabei eindeutig durch Adresse und Portnummer bestimmt.

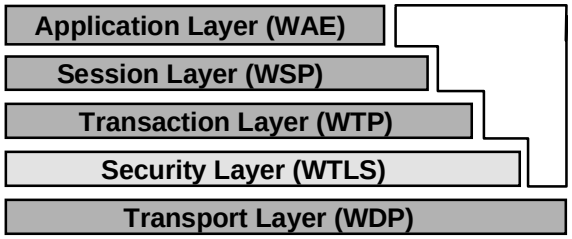
Gültige Portnummern sind z.B.: 9200 (Connectionless Session Service), 9202 (Secure Connectionless Session Service), 9201 (Session Service).





Wireless Transport Level Security

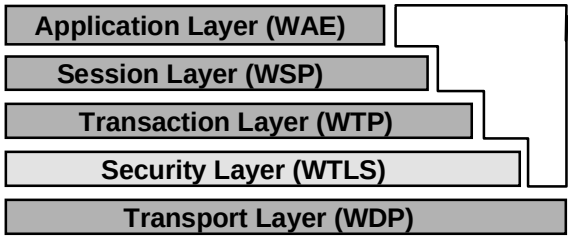
Funktionen des Security-Layers



Wireless Transport Level Security

Funktionen des Security-Layers

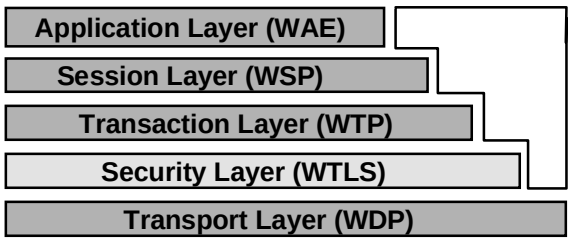
- **Digital Signing** — **mittels einer Einweg-Hash-Funktion (SHA oder MD5).**



Wireless Transport Level Security

Funktionen des Security-Layers

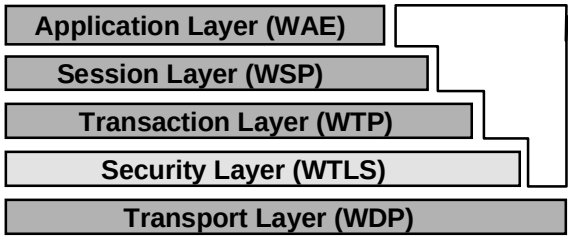
- **Digital Signing** — mittels einer Einweg-Hash-Funktion (SHA oder MD5).
- **Stream Cipher Encryption** — durch einfache XOR-Verknüpfung mit Daten aus einem Pseudo-Random-Stream.



Wireless Transport Level Security

Funktionen des Security-Layers

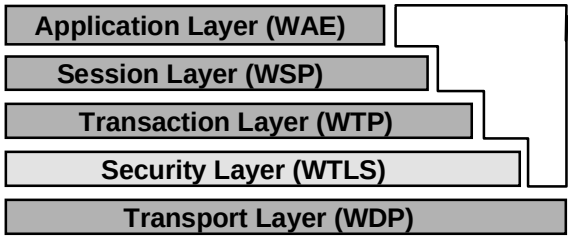
- **Digital Signing** — mittels einer Einweg-Hash-Funktion (SHA oder MD5).
- **Stream Cipher Encryption** — durch einfache XOR-Verknüpfung mit Daten aus einem Pseudo-Random-Stream.
- **Block Cipher Encryption** — mittels symmetrischer Verschlüsselungs-Algorithmen: IDEA, DES, Triple-DES.



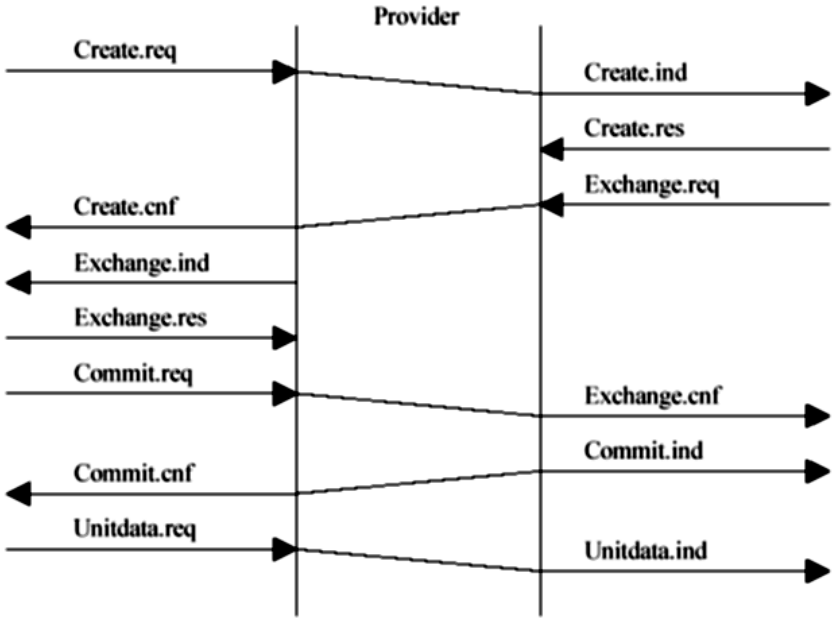
Wireless Transport Level Security

Funktionen des Security-Layers

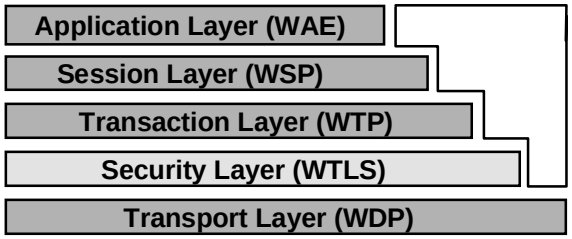
- **Digital Signing** — mittels einer Einweg-Hash-Funktion (SHA oder MD5).
- **Stream Cipher Encryption** — durch einfache XOR-Verknüpfung mit Daten aus einem Pseudo-Random-Stream.
- **Block Cipher Encryption** — mittels symmetrischer Verschlüsselungs-Algorithmen: IDEA, DES, Triple-DES.
- **Public Key Encryption** — mit den üblichen Standard-Verfahren: RSA/RC5-56.



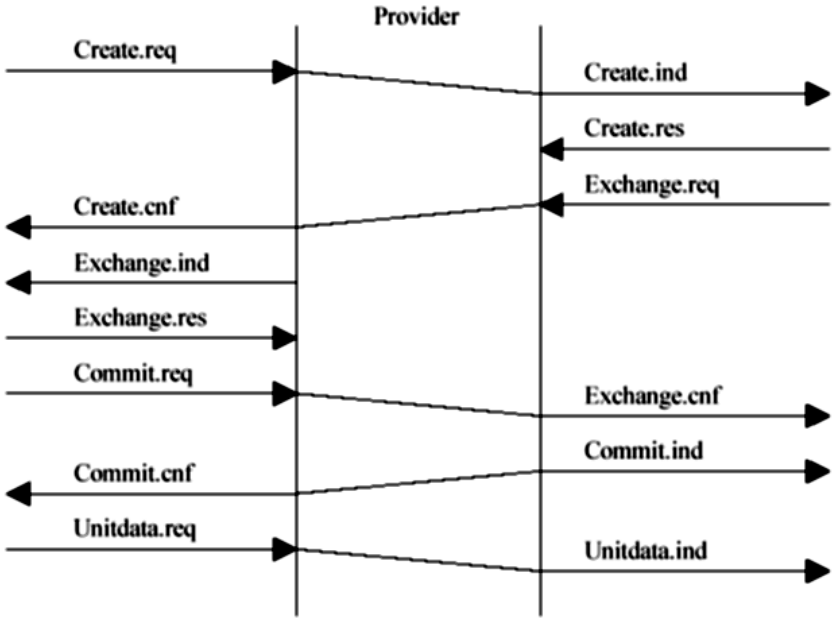
Wireless Transport Level Security



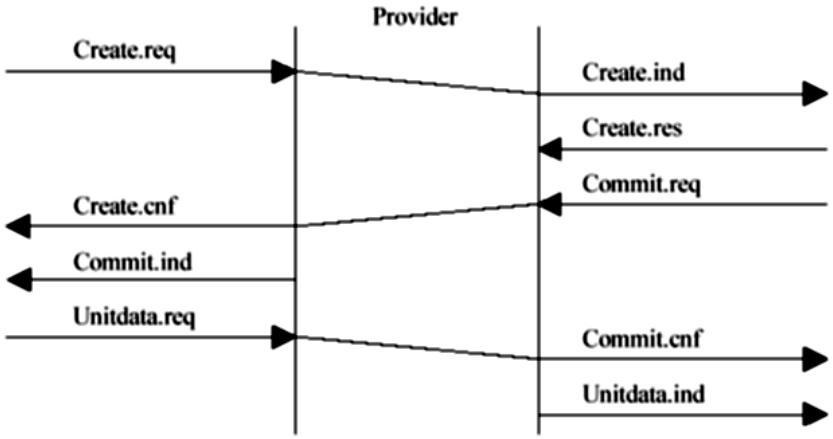
Volles und beschleunigtes Handshake

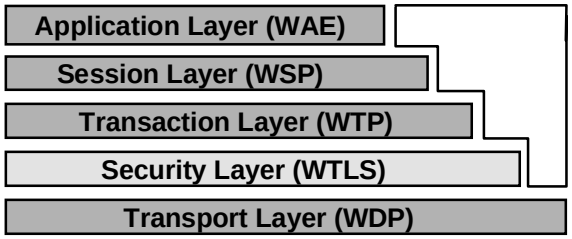


Wireless Transport Level Security



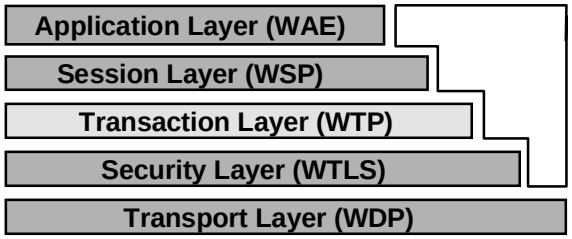
Volles und beschleunigtes Handshake





ireless ransacion Proocol

WTP ermöglicht Connection-freie, Packet-basierte Kommunikation durch verschiedene Transaction Services

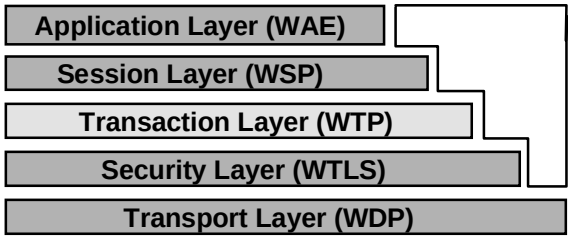


ireless ransacion Proocol

WTP ermöglicht Connection-freie, Packet-basierte Kommunikation durch verschiedene Transaction Services

Transaction Service Class 0:

Unreliable Invoke Message with not Response Message (Unreliable Datagram Service)



ireless ransacion Proocol

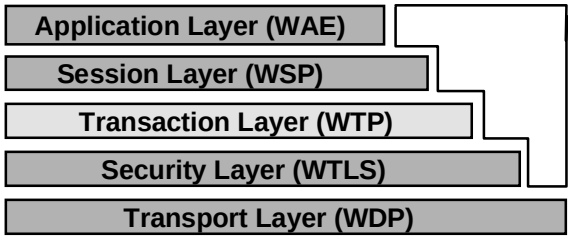
WTP ermöglicht Connection-freie, Packet-basierte Kommunikation durch verschiedene Transaction Services

Transaction Service Class 0:

Unreliable Invoke Message with not Response Message (Unreliable Datagram Service)

Transaction Service Class 1:

Reliable Invoke Message with not Response Message (Reliable Datagram Service)



ireless ransacion Proocol

WTP ermöglicht Connection-freie, Packet-basierte Kommunikation durch verschiedene Transaction Services

Transaction Service Class 0:

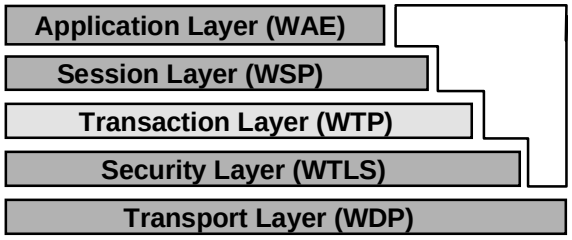
Unreliable Invoke Message with not Response Message (Unreliable Datagram Service)

Transaction Service Class 1:

Reliable Invoke Message with not Response Message (Reliable Datagram Service)

Transaction Service Class 2:

Reliable Invoke Message with exactly one Reliable Response Message (Invoke/Response Transaction Service)



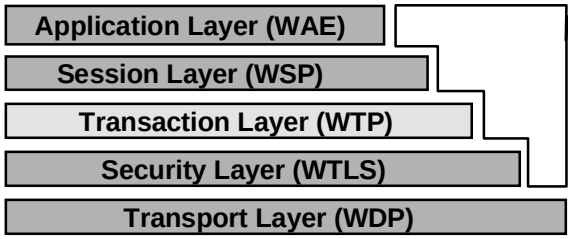
ireless ransacion Proocol

Übertragung der Daten in Protocol Data Units (PDU)

- **fester Header für oft benötigte Parameter**
- **variabler Header für seltener benötigte Parameter, realisiert durch Transport Information Items (TPI)**

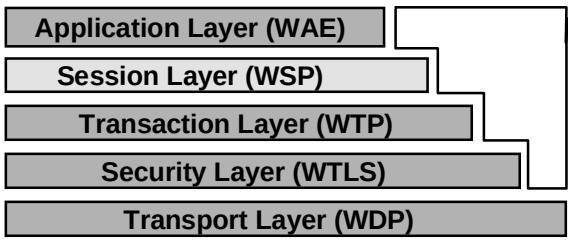
Beispiel: Invoke PDU (Nachricht senden)

Bit/Octet	0	1	2	3	4	5	6	7
1	CON	PDU Type = Invoke				GTR	TTR	RID
2 3	TID							
4	Version		TIDnew	U/P	RES	RES	TCL	



ireless ession Proocol

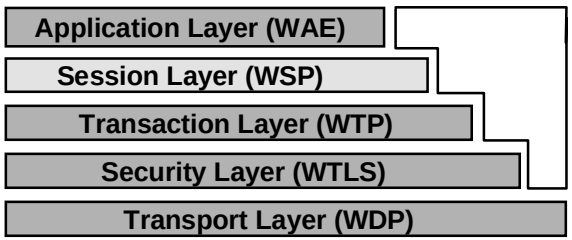
WSP basiert auf HTTP/1.1



ireless ession Proocol

WSP basiert auf HTTP/1.1

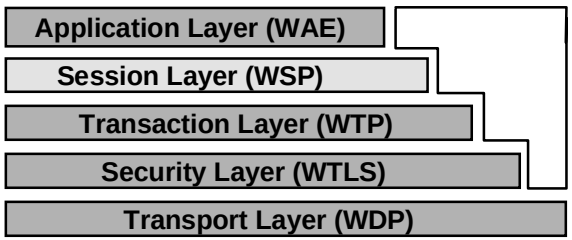
- Unterstützung der üblichen HTTP/1.1 Request/Reply-Methoden



ireless ession Proocol

WSP basiert auf HTTP/1.1

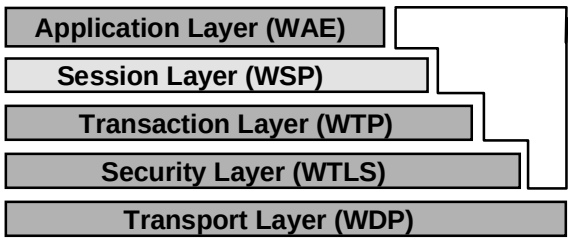
- Unterstützung der üblichen HTTP/1.1 Request/Reply-Methoden
- Unterstützung der üblichen HTTP/1.1 Request/Reply-Header



ireless ession Proocol

WSP basiert auf HTTP/1.1

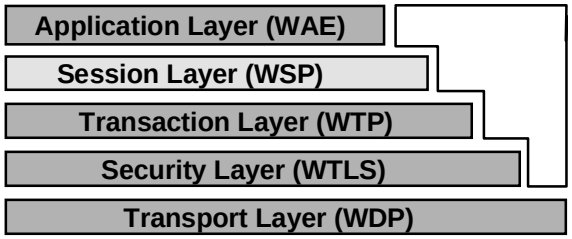
- Unterstützung der üblichen HTTP/1.1 Request/Reply-Methoden
- Unterstützung der üblichen HTTP/1.1 Request/Reply-Header
- Content typing (z.B. MIME)



ireless ession Proocol

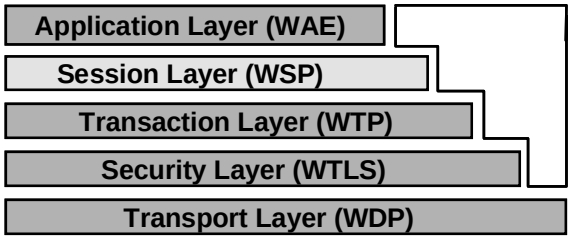
WSP basiert auf HTTP/1.1

- Unterstützung der üblichen HTTP/1.1 Request/Reply-Methoden
- Unterstützung der üblichen HTTP/1.1 Request/Reply-Header
- Content typing (z.B. MIME)
- Möglichkeit asynchroner Requests



ireless ession Proocol

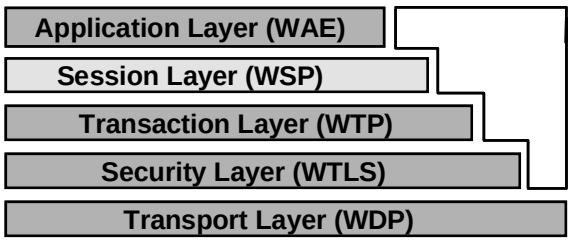
Erweiterungen gegenüber HTTP/1.1



ireless ession Proocol

Erweiterungen gegenüber HTTP/1.1

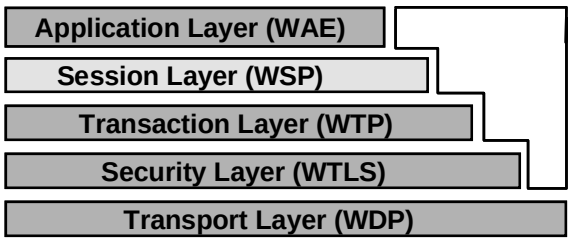
- **kompakte, binäre Header-Codierung**



ireless ession Proocol

Erweiterungen gegenüber HTTP/1.1

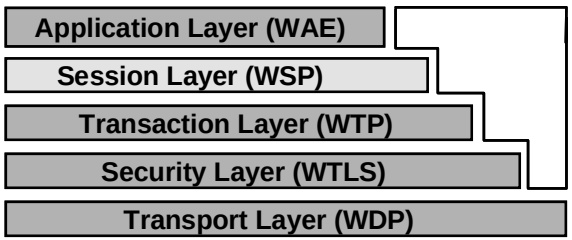
- **kompakte, binäre Header-Codierung**
- **Session Headers**



ireless ession Proocol

Erweiterungen gegenüber HTTP/1.1

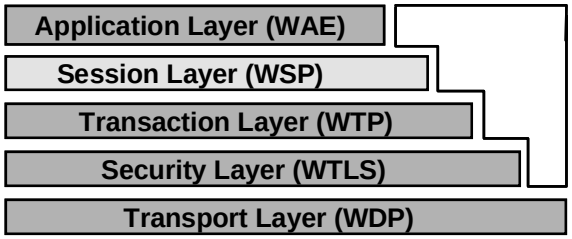
- **kompakte, binäre Header-Codierung**
- **Session Headers**
- **bestätigtes und unbestätigtes Data Push während einer Session**



ireless ession Proocol

Erweiterungen gegenüber HTTP/1.1

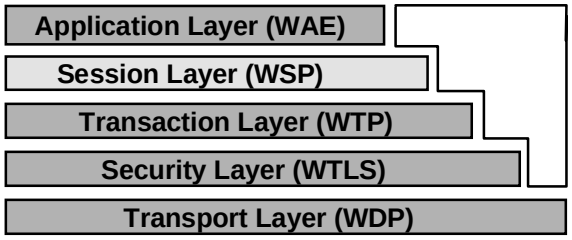
- kompakte, binäre Header-Codierung
- Session Headers
- bestätigtes und unbestätigtes Data Push während einer Session
- Suspend und Resume von Sessions



ireless ession Proocol

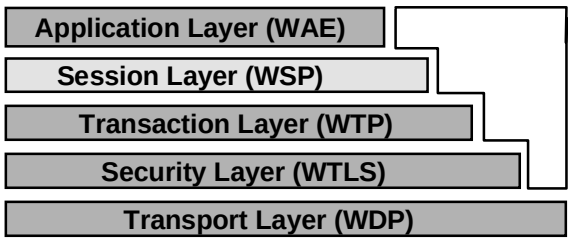
Erweiterungen gegenüber HTTP/1.1

- **kompakte, binäre Header-Codierung**
- **Session Headers**
- **bestätigtes und unbestätigtes Data Push während einer Session**
- **Suspend und Resume von Sessions**
- **Connectionless Services**



ireless ession Proocol

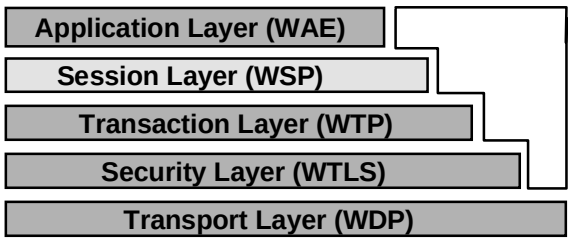
Connection-oriented vs Connection-less Services



ireless ession Proocol

Connection-oriented vs Connection-less Services

Connection-orientiert

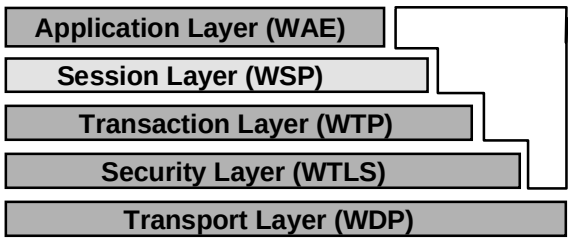


ireless ession Proocol

Connection-oriented vs Connection-less Services

Connection-orientiert

Pro: Hohe Zuverlässigkeit; es können Statusinformationen über die Session gespeichert werden (long-lived communication)



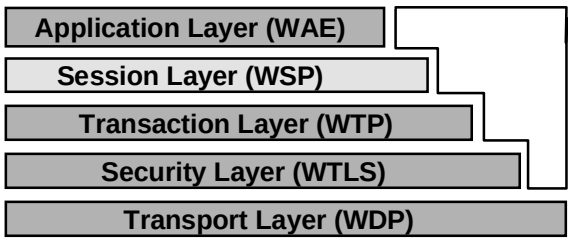
ireless ession Proocol

Connection-oriented vs Connection-less Services

Connection-orientiert

Pro: Hohe Zuverlässigkeit; es können Statusinformationen über die Session gespeichert werden (long-lived communication)

Contra: Zum Aufbau der Verbindung müssen viele Daten übertragen werden



ireless ession Proocol

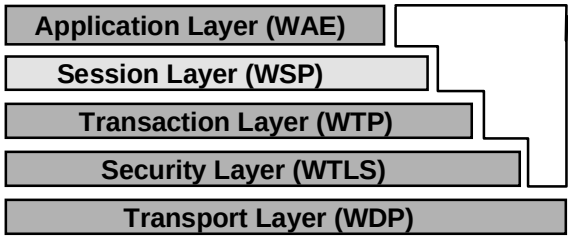
Connection-oriented vs Connection-less Services

Connection-orientiert

Pro: Hohe Zuverlässigkeit; es können Statusinformationen über die Session gespeichert werden (long-lived communication)

Contra: Zum Aufbau der Verbindung müssen viele Daten übertragen werden

Connection-frei



ireless ession Proocol

Connection-oriented vs Connection-less Services

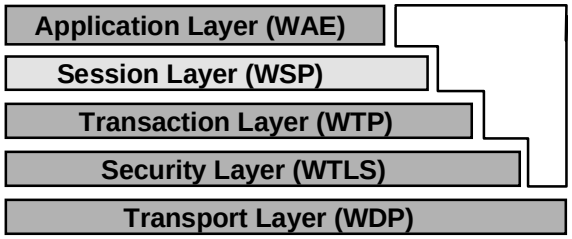
Connection-orientiert

Pro: Hohe Zuverlässigkeit; es können Statusinformationen über die Session gespeichert werden (long-lived communication)

Contra: Zum Aufbau der Verbindung müssen viele Daten übertragen werden

Connection-frei

Pro: schnell, kein Session Creation Overhead, kein Reliability Overhead



ireless ession Proocol

Connection-oriented vs Connection-less Services

Connection-orientiert

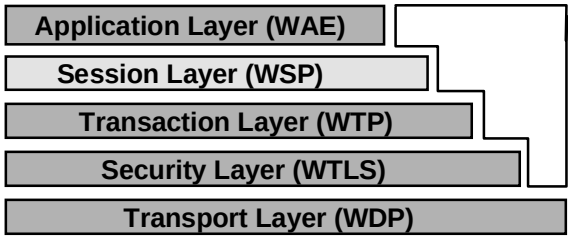
Pro: Hohe Zuverlässigkeit; es können Statusinformationen über die Session gespeichert werden (long-lived communication)

Contra: Zum Aufbau der Verbindung müssen viele Daten übertragen werden

Connection-frei

Pro: schnell, kein Session Creation Overhead, kein Reliability Overhead

Contra: keine Statusinformationen, geringere Zuverlässigkeit

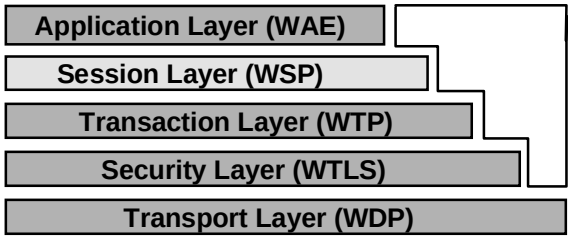


ireless ession Proocol

Die wichtigsten Service Primitives

Primitive / Parameter	S-Connect			
	req	*Ind*	*Res*	*cnf*
Server Address	M	M(=)	–	–
Client Address	M	M(=)	–	–
Client Headers	O	C(=)	–	–
Requested Capabilities	O	M	–	–
Server Headers	–	–	O	C(=)
Negotiated Capabilities	–	–	O	M(=)

S-Connect zum Aufbau einer Session

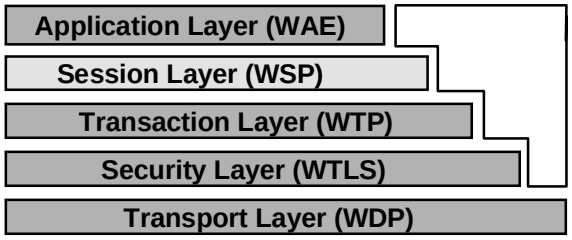


ireless ession Proocol

Die wichtigsten Service Primitives

Parameter \ Primitive	S-MethodInvoke			
	req	*ind*	*res*	*cnf*
Client Transaction Id	M	–	–	M(=)
Server Transaction Id	–	M	M(=)	–
Method	M	M(=)	–	–
Request URI	M	M(=)	–	–
Request Headers	O	C(=)	–	–
Request Body	C	C(=)	–	–

S-MethodInvoke zum Starten eines Requests

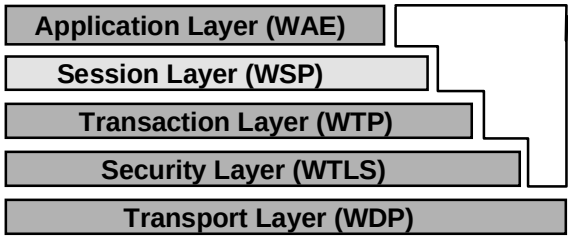


ireless ession Proocol

Die wichtigsten Service Primitives

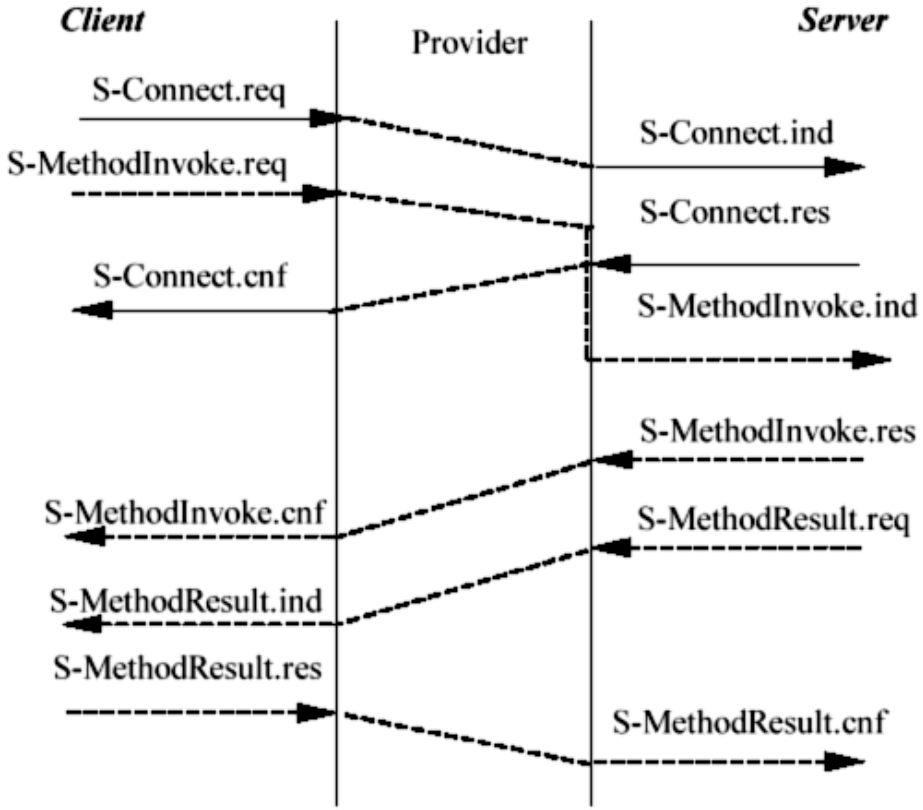
Parameter \ Primitive	S-MethodResult			
	req	*ind*	*res*	*cnf*
Server Transaction Id	M	–	–	M(=)
Client Transaction Id	–	M	M(=)	–
Status	M	M(=)	–	–
Response Headers	O	C(=)	–	–
Response Body	C	C(=)	–	–
Acknowledgement Headers	–	–	O	P(=)

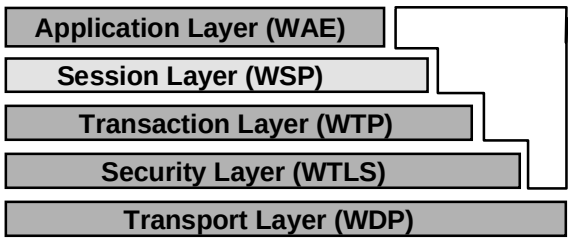
S-MethodResult zum Senden des Replys



ireless ession Proocol

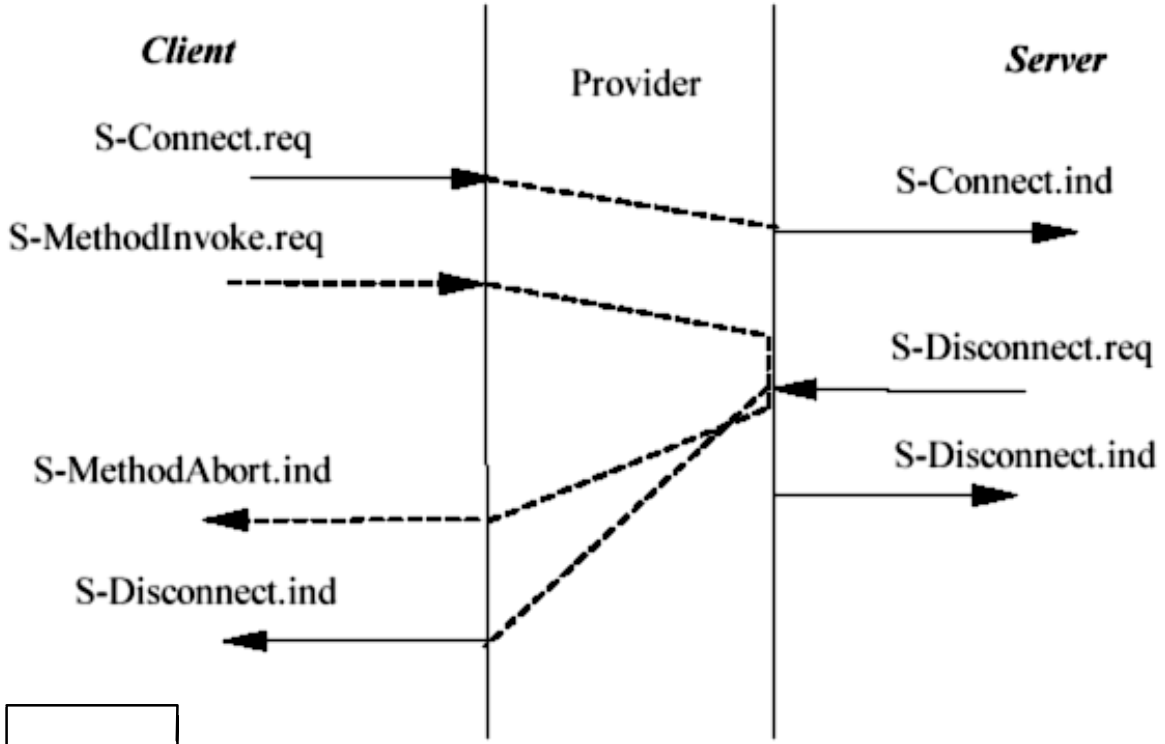
Beispiel 1: Successful Session Establishment

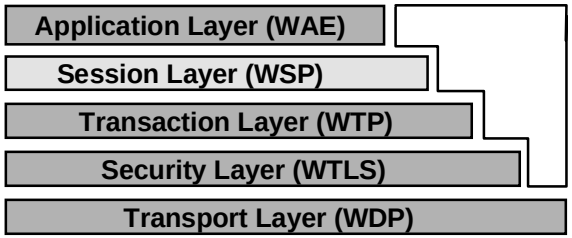




ireless ession Proocol

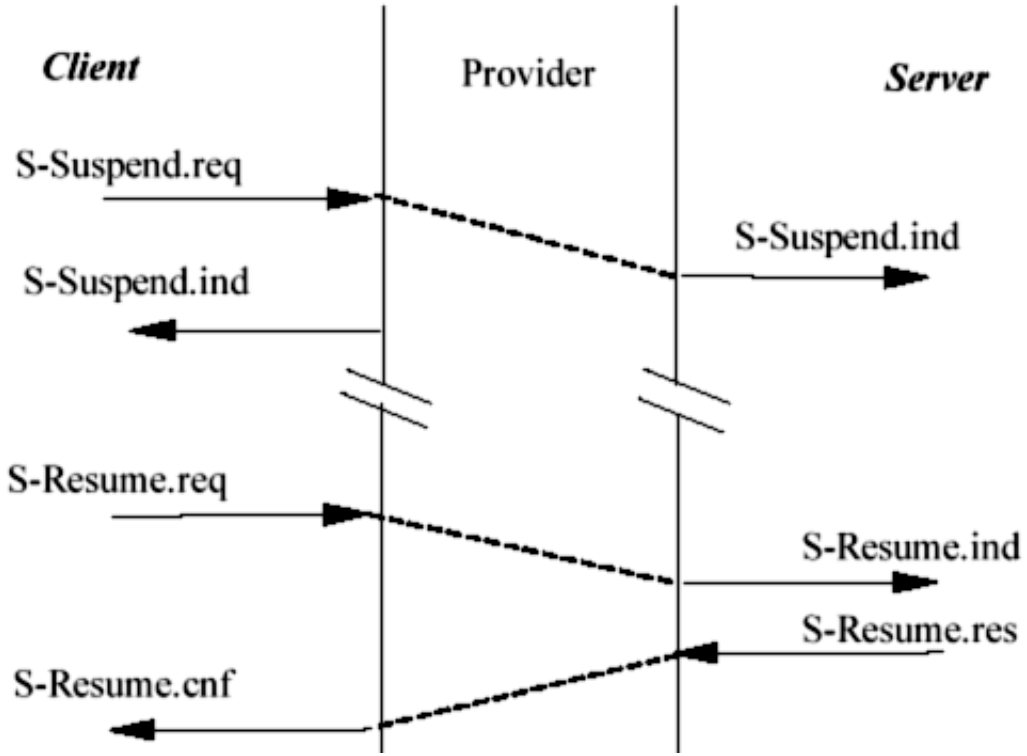
Beispiel 2: Refused Session Establishment

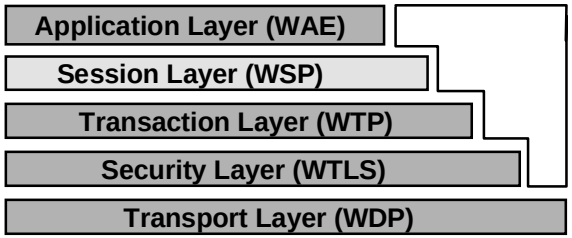




ireless ession Proocol

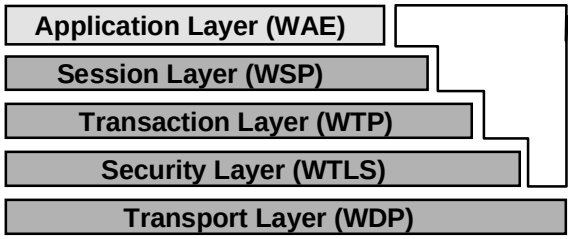
Beispiel 3: Session Suspension and Resume





ireless pplicaion nironen

WAP - und damit der WAE-Microbrowser - unterscheidet zwischen verschiedenen Content-Types



ireless pplicaion nironen

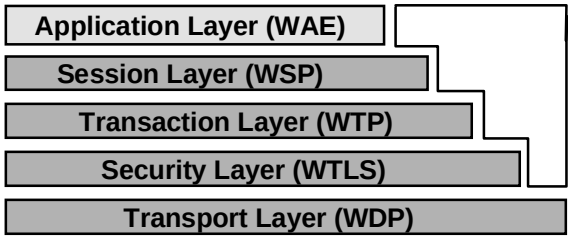
WAP - und damit der WAE-Microbrowser - unterscheidet zwischen verschiedenen Content-Types

1. **Wireless Markup Language (WML)**

   text/wml (.wml)

Zum Darstellen textueller Inhalte durch sog. Decks und Cards. WML basiert auf XML und ist daher HTML sehr ähnlich.

WML-Dokumente können Bilder enthalten.



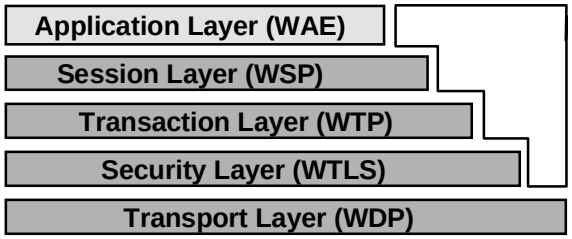
ireless pplicaion nironen

WAP - und damit der WAE-Microbrowser - unterscheidet zwischen verschiedenen Content-Types

2. Bitmap (WBMP)

image/x-wap.wbmp

Mit WML-Bitmaps werden grafische Inhalte innerhalb von WML-Decks dargestellt. Momentan ist die Farbtiefe auf 1 Bit beschränkt (nicht zuletzt wegen der Displays), das Format ist jedoch beliebig erweiterbar, z.B. auf TrueColor. Die Bilddaten können raw oder komprimiert vorliegen.



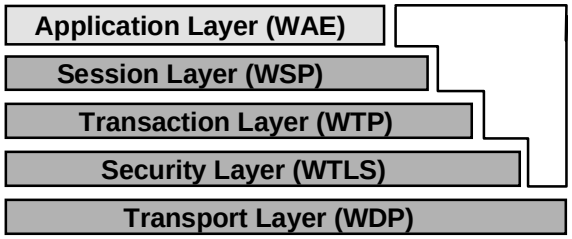
ireless pplicaion nironen

WAP - und damit der WAE-Microbrowser - unterscheidet zwischen verschiedenen Content-Types

3. vCard (Business-Card)

text/x-vcard (.vcf)

Die WAP-vCard ist die Umsetzung eines bereits bestehenden Industrie-Standards zum Austausch von Telefonbucheinträgen und Business Cards. Der Austausch erfolgt in der Regel durch einfache Datagramme.



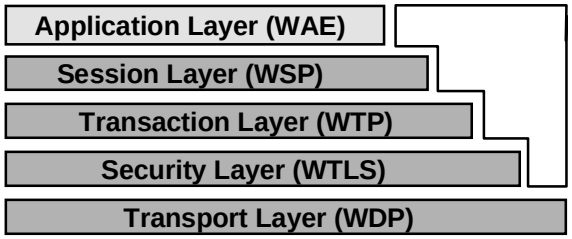
ireless pplicaion nironen

WAP - und damit der WAE-Microbrowser - unterscheidet zwischen verschiedenen Content-Types

4. vCalendar (Kalender-Item)

text/x-vcal (.vcs)

Wie die vCard ist vCalendar ein bestehender Industrie-Standard, diesmal zum Austausch von Termin-Informationen. Auch hier geschieht die Übertragung durch Datagramme (WDP).



ireless arkup anguage

ireless arkup anguage

Grundgerüst eines WML-Dokuments

```
<?xml version="1.0"?>
<!DOCTYPE WML PUBLIC "-//WAPFORUM//DTD WML 1.0//EN"
 "http://www.wapforum.org/DTD/wml.xml">
<WML>
  <CARD>
    ...
  </CARD>
</WML>
```

ireless arkup anguage

Beispiel für eine WML-Card

```
<CARD NAME="eCard">
  <DO TYPE="ACCEPT">
    <GO URL="/submit?N=$(N)&S=$(S)"/>
  </DO>
  Enter name: <INPUT KEY="N"/>
  Choose speed:
  <SELECT KEY="S">
    <OPTION VALUE="0">Fast</OPTION>
    <OPTION VALUE="1">Slow</OPTION>
  </SELECT>
</CARD>
```

ireless arkup anguage

Beispiel für eine WML-Card

```
<CARD NAME="eCard">
  <DO TYPE="ACCEPT">
    <GO URL="/submit?N=$(N)&S=$(S)"/>
  </DO>
  Enter name: <INPUT KEY="N"/>
  Choose speed:
  <SELECT KEY="S">
    <OPTION VALUE="0">Fast</OPTION>
    <OPTION VALUE="1">Slow</OPTION>
  </SELECT>
</CARD>
```

Das Pendant in HTML

```
<FORM NAME="eCard" ACTION="/submit" METHOD="POST">
  Enter name: <INPUT TYPE=TEXT NAME="N"><BR>
  Choose Speed:
  <SELECT NAME="S">
    <OPTION VALUE="0">Fast
    <OPTION VALUE="1">Slow
  </SELECT>
</FORM>
```

ireless arkup anguage

Um Zeit und Speicherplatz zu sparen, liegen WML-Decks nicht als Plaintext, sondern Byte-codiert vor.

ireless arkup anguage

Um Zeit und Speicherplatz zu sparen, liegen WML-Decks nicht als Plaintext, sondern Byte-codiert vor.

Beispiel für WML-Bytecode-Darstellung:

WML-Deck in Plaintext

```
<wml><card id="abc" ordered="true">
  <p>
    <do type="accept">
      <go href="http://xyz.org/s"/>
    </do>
    X: $(X) <br/>
    Y: $(&#x59;) <br/>
    Enter name: <input type="text" name="N"/>
  </p>
</card></wml>
```

***WML-Bytecode* (in Hex-Darstellg.)**

```
01 04 6A 04 'X 00 'Y 00
7F E7 21 03 'a 'b 'c 00
33 01 20 E8 38 01 AB 4B
03 'x 'y 'z 00 88 03 's
00 01 03 '  'X ': '  00
82 00 26 03 '  'Y ': '
00 82 02 28 03 '  'E 'n
'n 't 'e 'r '  'n 'a 'm
'e ': '  00 AF 48 18 03
'N 00 01 01 01
```

ireless arkup anguage

Übersetzungstabelle

01	XML Version 1.1	AB	<go>	82	Variablen-Referenz
04	WML 1.1 Public ID	4B	href="http://"	02	Variablen-Offset 2
6A	Charset UTF-8	03	Inline-String folgt	26	
04	Länge der Stringtable	"xyz" 00	Inline-String	03	Inline-String folgt
"X" 00 "Y" 00	Stringtable	88	".org/"	" Enter name: " 00	Inline-String
7F	<wml>	03	Inline-String folgt	AF	<input>
E7	<card>	"s" 00	Inline-String	48	type="text"
55	id=	01	Ende (von <go>)	21	name=
03	Inline-String folgt	01	Ende (von <do>)	03	Inline-String folgt
"abc" 00	Inline-String	03	Inline-String folgt	"N" 00	Inline-String
33	ordered="true"	" X: " 00	Inline-String	01	Ende (der Input-Attribute)
01	Ende (der <card>-Attribute)	82	Variablen-Referenz	01	Ende (von <p>)
60	<p>	00	Variablen-Offset 0	01	Ende (von <card>)
E8	<do>	26	 	01	Ende (von <wml>)
38	type="accept"	03	Inline-String folgt		
01	Ende (der <do>-Attribute)	" Y: " 00	Inline-String		

ireless arkup anguage

Darstellung von WML-Dokumenten

Beispiel-Deck

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
                "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="first" title="Demo Application">
    <do type="accept">
      <go href="#next"/>
    </do>
    <p>
      <br/>Druecken Sie "YES", um fort- zusetzen.
    </p>
  </card>
  <card id="next" title="Willkommen!">
    <p>
      Vorname:  <input type="text" name="first"/>
      Nachname: <input type="text" name="last"/>
    </p>
  </card>
</wml>
```

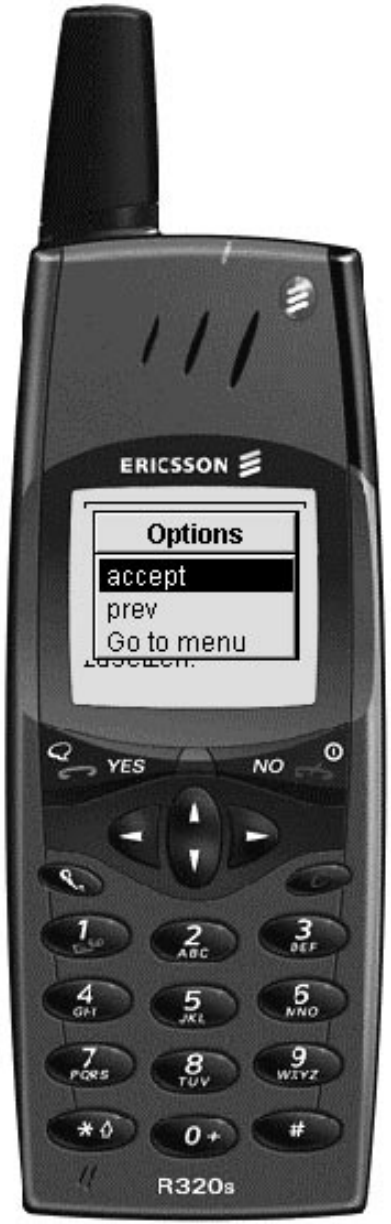
ireless arkup anguage

Darstellung von WML-Dokumenten

Beispiel-Deck

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
                "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="first" title="Demo Application">
    <do type="accept">
      <go href="#next"/>
    </do>
    <p>
      <br/>Druecken Sie "YES", um fort- zusetzen.
    </p>
  </card>
  <card id="next" title="Willkommen!">
    <p>
      Vorname:  <input type="text" name="first"/>
      Nachname: <input type="text" name="last"/>
    </p>
  </card>
</wml>
```

ireless arkup anguage



Darstellung von WML-Dokumenten

Beispiel-Deck

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
                "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="first" title="Demo Application">
    <do type="accept">
      <go href="#next"/>
    </do>
    <p>
      <br/>Druecken Sie "YES", um fort- zusetzen.
    </p>
  </card>
  <card id="next" title="Willkommen!">
    <p>
      Vorname:  <input type="text" name="first"/>
      Nachname: <input type="text" name="last"/>
    </p>
  </card>
</wml>
```

ireless arkup anguage

Darstellung von WML-Dokumenten

Beispiel-Deck

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
                "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="first" title="Demo Application">
    <do type="accept">
      <go href="#next"/>
    </do>
    <p>
      <br/>Druecken Sie "YES", um fort- zusetzen.
    </p>
  </card>
  <card id="next" title="Willkommen!">
    <p>
      Vorname:  <input type="text" name="first"/>
      Nachname: <input type="text" name="last"/>
    </p>
  </card>
</wml>
```

crip

- **entwickelt nach dem Vorbild von JavaScript;**
- **Ziel: sinnvolle Erweiterung von WML und bessere Nutzung der zur Verfügung stehenden Bandbreite;**
- **WMLScript ist Byte-codiert und wird von einer Virtual Machine interpretiert.**

crip

Die Standard-Libraries von WMLScript umfassen:

- **oft benötigte mathematische Funktionen**
- **grundlegende Funktionen zur String-Verarbeitung**
- **URL-bezogene Funktionen**
- **Funktionen zum Browser-Interface**
- **einfache Dialog-Interfaces**
- **Software-Gleitkomma-Arithmetik**

crip

Beispiel für WMLScript-Code

```
function currencyConvertor(currency, exchRate)
{
  return currency * exchRate;
}
function myDay(sunShines) {
  var result;
  var myStr = "Not So Good";
  if (sunShines) {
    result = String.subString(myStr, 7, 4);
  } else {
    result = myStr;
  };
  return result;
}
```

crip

Beispiel für WMLScript-Code:

```
function currencyConvertor(currency, exchRate) {
  return currency * exchRate;
}
function myDay(sunShines) {
  var result;
  var myStr = "Not So Good";
  if (sunShines) {
    result = String.subString(myStr, 7, 4);
  } else {
    result = myStr;
  };
  return result;
}
```

Nur minimale Unterschiede!

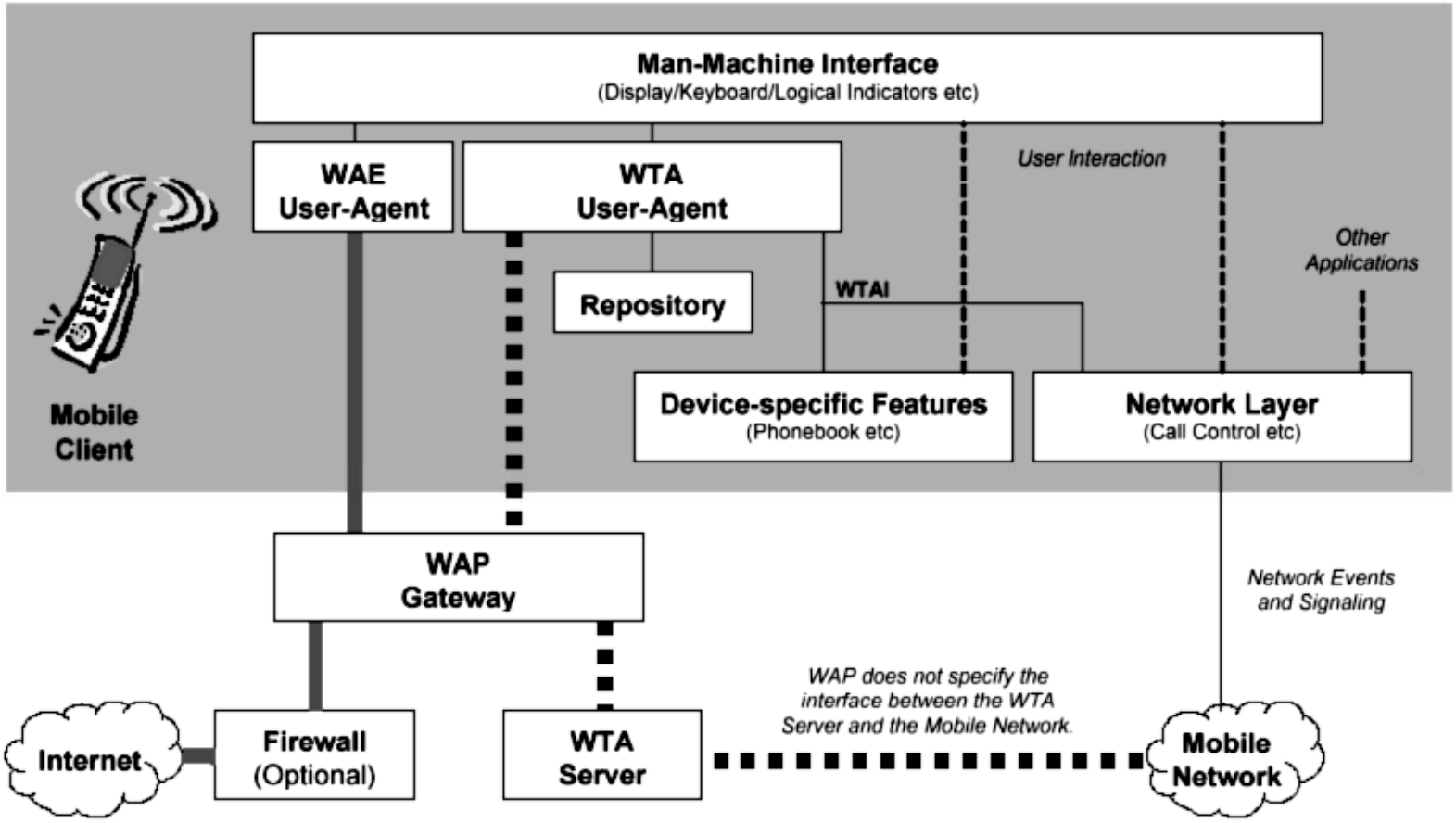
zum Beispiel bei Methodenaufrufen

Das JavaScript-Äquivalent:

```
function currencyConvertor(currency, exchRate) {
  return currency * exchRate;
}

function myDay(sunShines) {
  var result;
  var myStr = "Not So Good";
  if (sunShines) {
    result = myStr.substr(7, 4);
  } else {
    result = myStr;
  }
  return result;
}
```

Wireless Telephony Application (Interface)



eine mögliche WTA-Konfiguration

Wireless Telephony Application (Interface)

WTA Permission Classes

Wireless Telephony Application (Interface)

WTA Permission Classes

Blanket Permission:

Der Benutzer gibt der jeweiligen WTAI-Funktion eine Blanko-Erlaubnis, üblicherweise zum Zeitpunkt ihrer Einrichtung. Die Permission kann ihr jedoch jederzeit wieder entzogen werden.

Wireless Telephony Application (Interface)

WTA Permission Classes

Blanket Permission:

Der Benutzer gibt der jeweiligen WTAI-Funktion eine Blanko-Erlaubnis, üblicherweise zum Zeitpunkt ihrer Einrichtung. Die Permission kann ihr jedoch jederzeit wieder entzogen werden.

Session Permission:

Der Benutzer gibt die Permission für die Dauer einer bestimmten Session. Die Permission geht verloren, sobald der Benutzer sie aufhebt, spätestens aber bei Beendigung der Session.

Wireless Telephony Application (Interface)

WTA Permission Classes

Blanket Permission:

Der Benutzer gibt der jeweiligen WTAI-Funktion eine Blanko-Erlaubnis, üblicherweise zum Zeitpunkt ihrer Einrichtung. Die Permission kann ihr jedoch jederzeit wieder entzogen werden.

Session Permission:

Der Benutzer gibt die Permission für die Dauer einer bestimmten Session. Die Permission geht verloren, sobald der Benutzer sie aufhebt, spätestens aber bei Beendigung der Session.

Single Action Permission:

Die Permission gilt nur für einen einzigen Aufruf der Funktion. Wird die Funktion später erneut aufgerufen, muss die Permission neu vergeben werden.

ieraurereichnis

[1] WAP Forum, 1999: "Wireless Application Protocol Architecure Overview"

[2] Flaherty/Dahm, 1998: "Wireless Application Protocol Technical Overview"

[3] WAP Forum, 1999: "Wireless Markup Language Specification"

[4] WAP Forum, 1999: "Wireless Telephony Application Specification"

[5] WAP Forum, 1999: "Wireless Session Protocol Specification"

[6] WAP Forum, 1999: "Wireless Transaction Protocol Specification"

[7] WAP Forum, 1999: "Wireless Transport Layer Security Specification"

[8] WAP Forum, 1999: "WMLScript Language Specification"

[9] WAP Forum, 1999: "Wireless Datagram Protocol Specification"